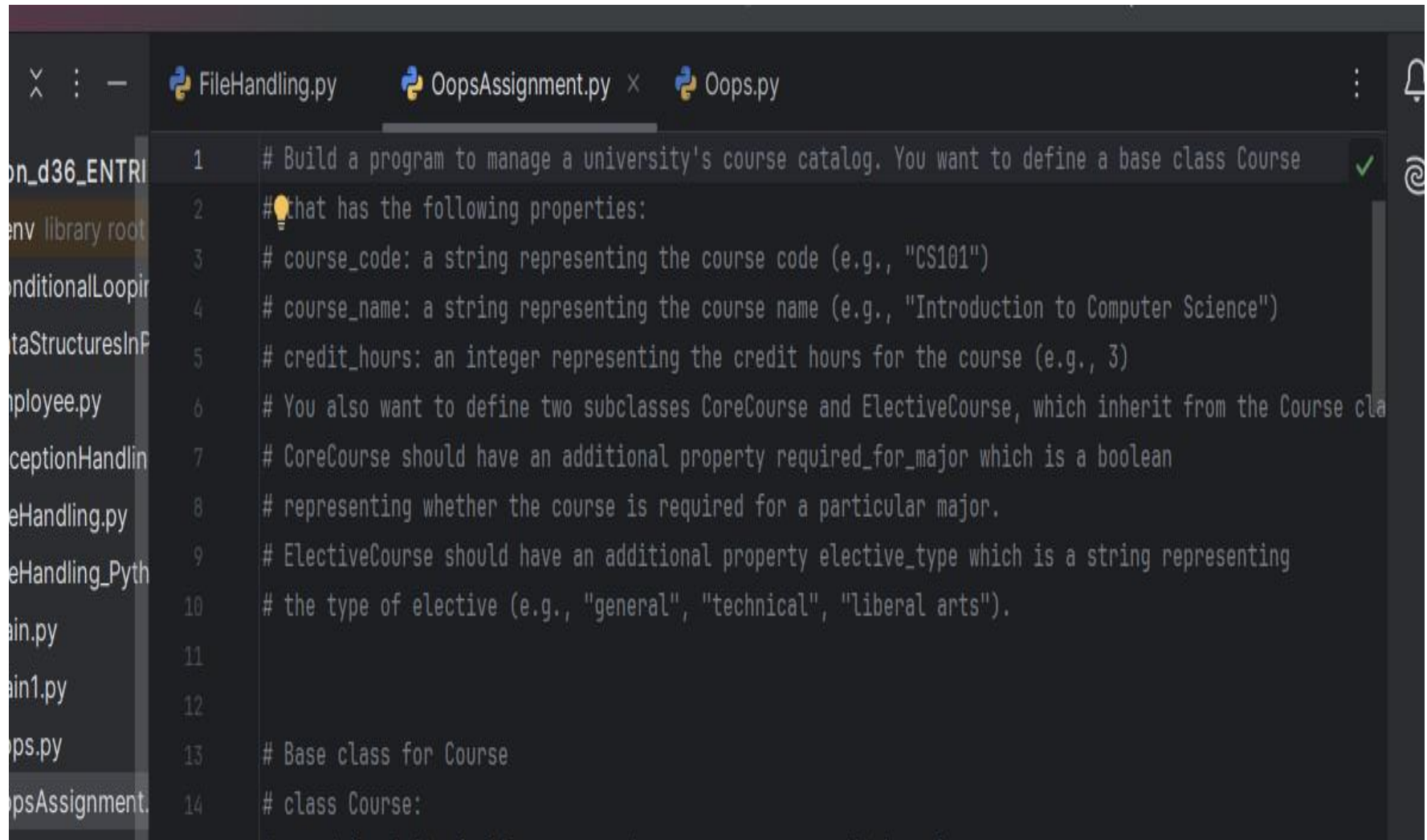


ASSIGNMENT ON OOPS IN PYTHON

BY
SHINO MARY PHILIPOSE

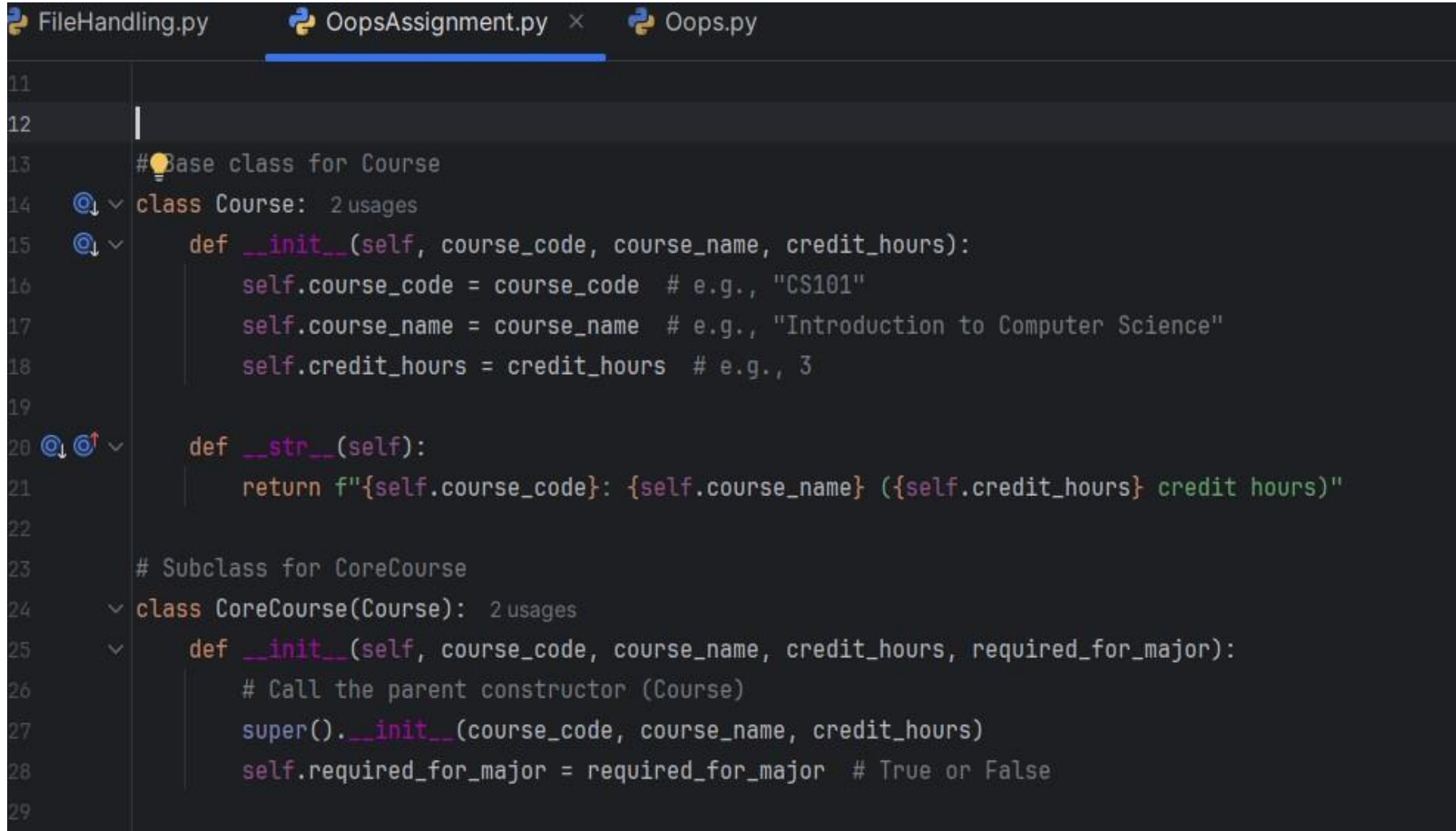
Q.1



```
FileHandling.py  OopsAssignment.py  Oops.py

1  # Build a program to manage a university's course catalog. You want to define a base class Course
2  # that has the following properties:
3  # course_code: a string representing the course code (e.g., "CS101")
4  # course_name: a string representing the course name (e.g., "Introduction to Computer Science")
5  # credit_hours: an integer representing the credit hours for the course (e.g., 3)
6  # You also want to define two subclasses CoreCourse and ElectiveCourse, which inherit from the Course class
7  # CoreCourse should have an additional property required_for_major which is a boolean
8  # representing whether the course is required for a particular major.
9  # ElectiveCourse should have an additional property elective_type which is a string representing
10 # the type of elective (e.g., "general", "technical", "liberal arts").
11
12
13 # Base class for Course
14 # class Course:
```

Q.1



```
FileHandling.py  OopsAssignment.py  Oops.py

11
12
13 # Base class for Course
14 class Course:
15     def __init__(self, course_code, course_name, credit_hours):
16         self.course_code = course_code # e.g., "CS101"
17         self.course_name = course_name # e.g., "Introduction to Computer Science"
18         self.credit_hours = credit_hours # e.g., 3
19
20     def __str__(self):
21         return f"{self.course_code}: {self.course_name} ({self.credit_hours} credit hours)"
22
23 # Subclass for CoreCourse
24 class CoreCourse(Course):
25     def __init__(self, course_code, course_name, credit_hours, required_for_major):
26         # Call the parent constructor (Course)
27         super().__init__(course_code, course_name, credit_hours)
28         self.required_for_major = required_for_major # True or False
29
```

Q.1

```
FileHandling.py  OopsAssignment.py  Oops.py

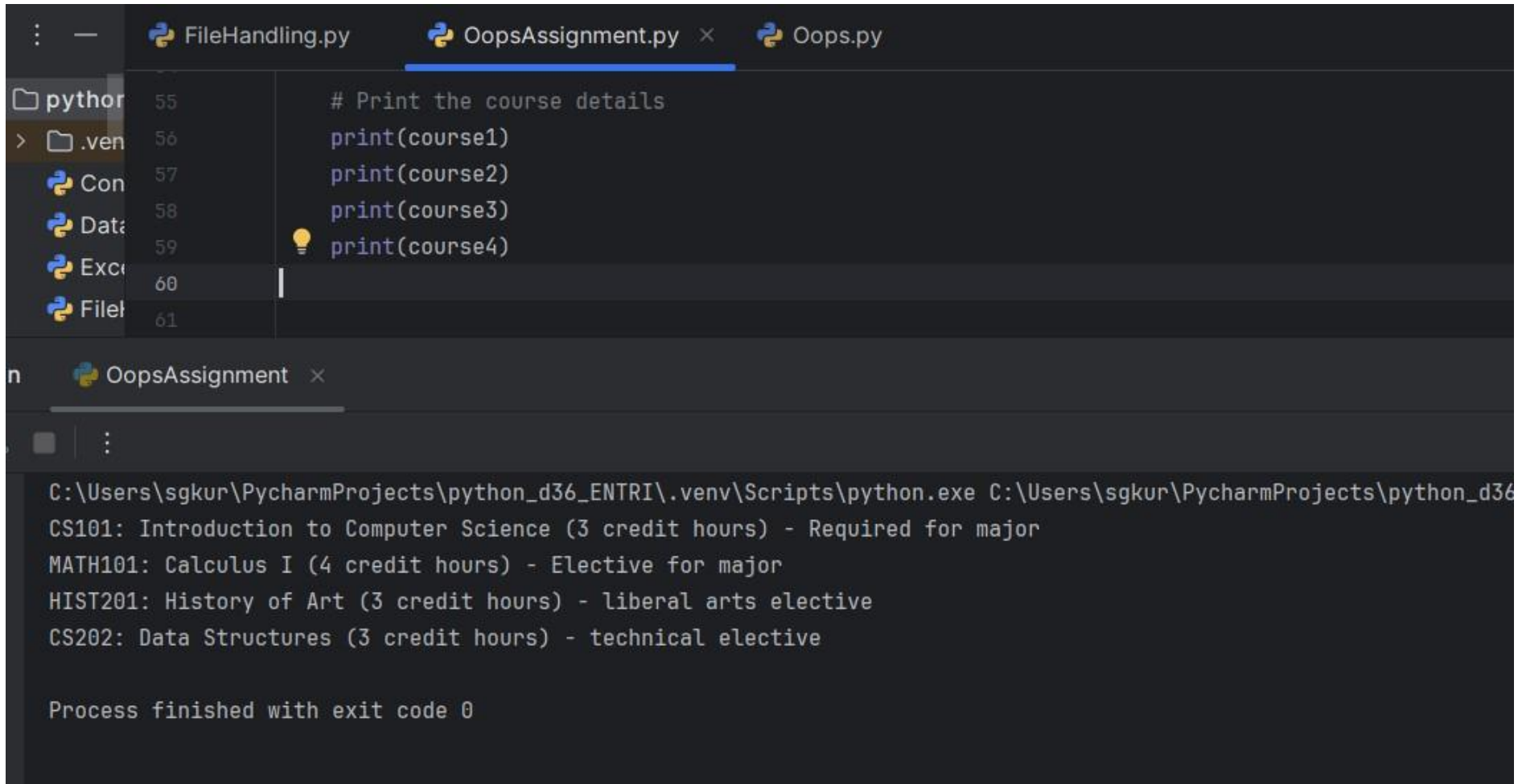
24  class CoreCourse(Course): 2 usages
29
30  def __str__(self):
31      required_status = "Required" if self.required_for_major else "Elective"
32      return f"{super().__str__()} - {required_status} for major"
33
34  # Subclass for ElectiveCourse
35  class ElectiveCourse(Course): 2 usages
36      def __init__(self, course_code, course_name, credit_hours, elective_type):
37          # Call the parent constructor (Course)
38          super().__init__(course_code, course_name, credit_hours)
39          self.elective_type = elective_type # e.g., "general", "technical", "liberal arts"
40
41  def __str__(self):
42      return f"{super().__str__()} - {self.elective_type} elective"
43
44  # Example usage
45  if __name__ == "__main__":
46      # Create some courses
```

Q.1

```
FileHandling.py  OopsAssignment.py  ×  Oops.py

42         return f"{super().__str__()} - {self.elective_type} elective"
43
44     # Example usage
45     if __name__ == "__main__":
46         # Create some courses
47         course1 = CoreCourse(course_code="CS101", course_name="Introduction to Computer Science", credit_hours=3,
48                               required_for_major=True)
49         course2 = CoreCourse(course_code="MATH101", course_name="Calculus I", credit_hours=4, required_for_major=False)
50         course3 = ElectiveCourse(course_code="HIST201", course_name="History of Art", credit_hours=3,
51                                   elective_type="liberal arts")
52         course4 = ElectiveCourse(course_code="CS202", course_name="Data Structures", credit_hours=3,
53                                   elective_type="technical")
54
55         # Print the course details
56         print(course1)
57         print(course2)
58         print(course3)
59         print(course4)
60
```

Q.1



The screenshot shows the PyCharm IDE interface. The top toolbar contains icons for FileHandling.py, OopsAssignment.py (selected), and Oops.py. The left sidebar shows a file explorer with a tree view containing 'python' and '.venv'. The main editor window displays the code in 'OopsAssignment.py'.

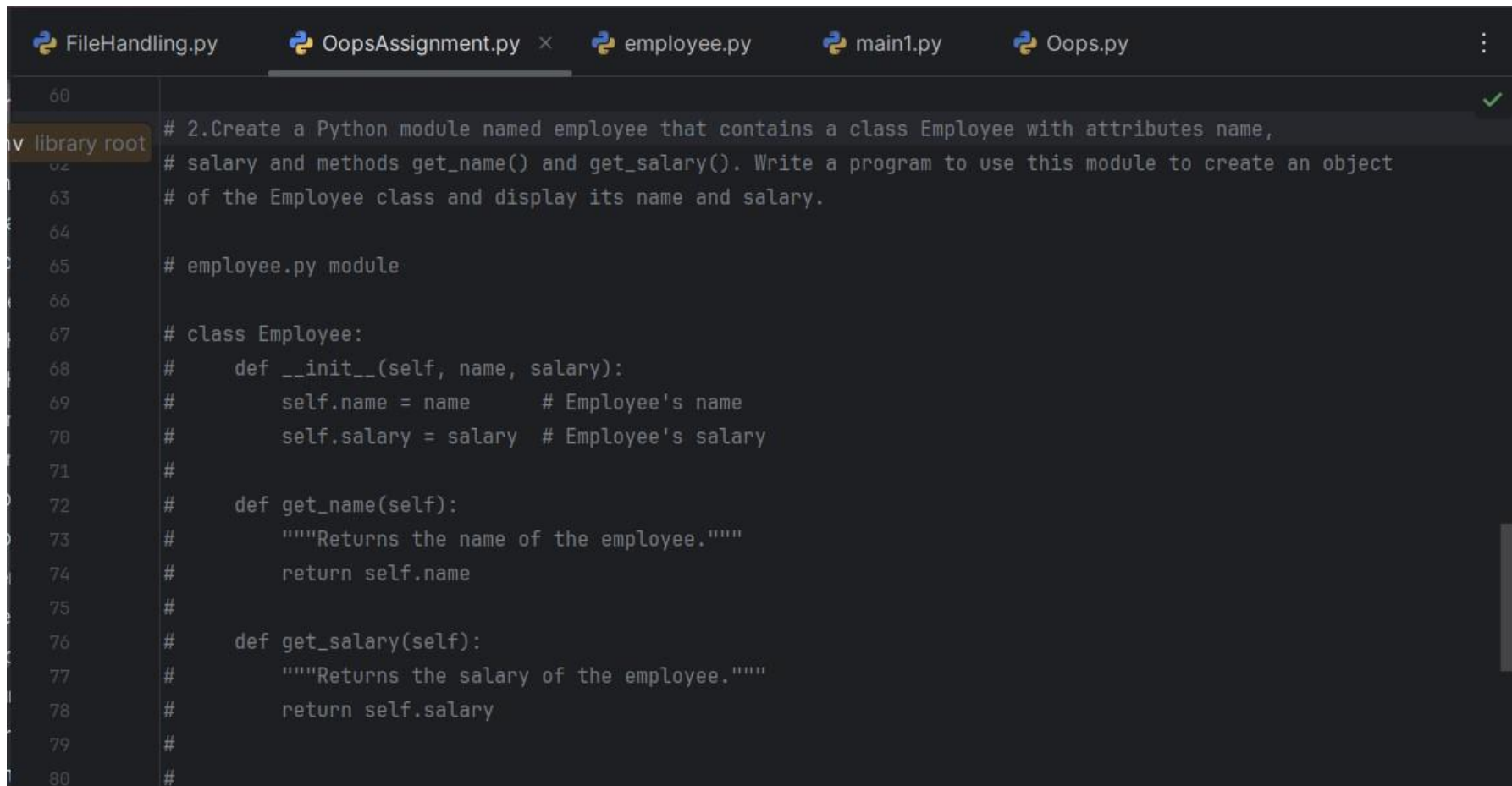
```
55 # Print the course details
56 print(course1)
57 print(course2)
58 print(course3)
59 print(course4)
60
61
```

The bottom console window shows the execution output:

```
C:\Users\sgkur\PycharmProjects\python_d36_ENTRI\.venv\Scripts\python.exe C:\Users\sgkur\PycharmProjects\python_d36
CS101: Introduction to Computer Science (3 credit hours) - Required for major
MATH101: Calculus I (4 credit hours) - Elective for major
HIST201: History of Art (3 credit hours) - liberal arts elective
CS202: Data Structures (3 credit hours) - technical elective

Process finished with exit code 0
```

Q.2

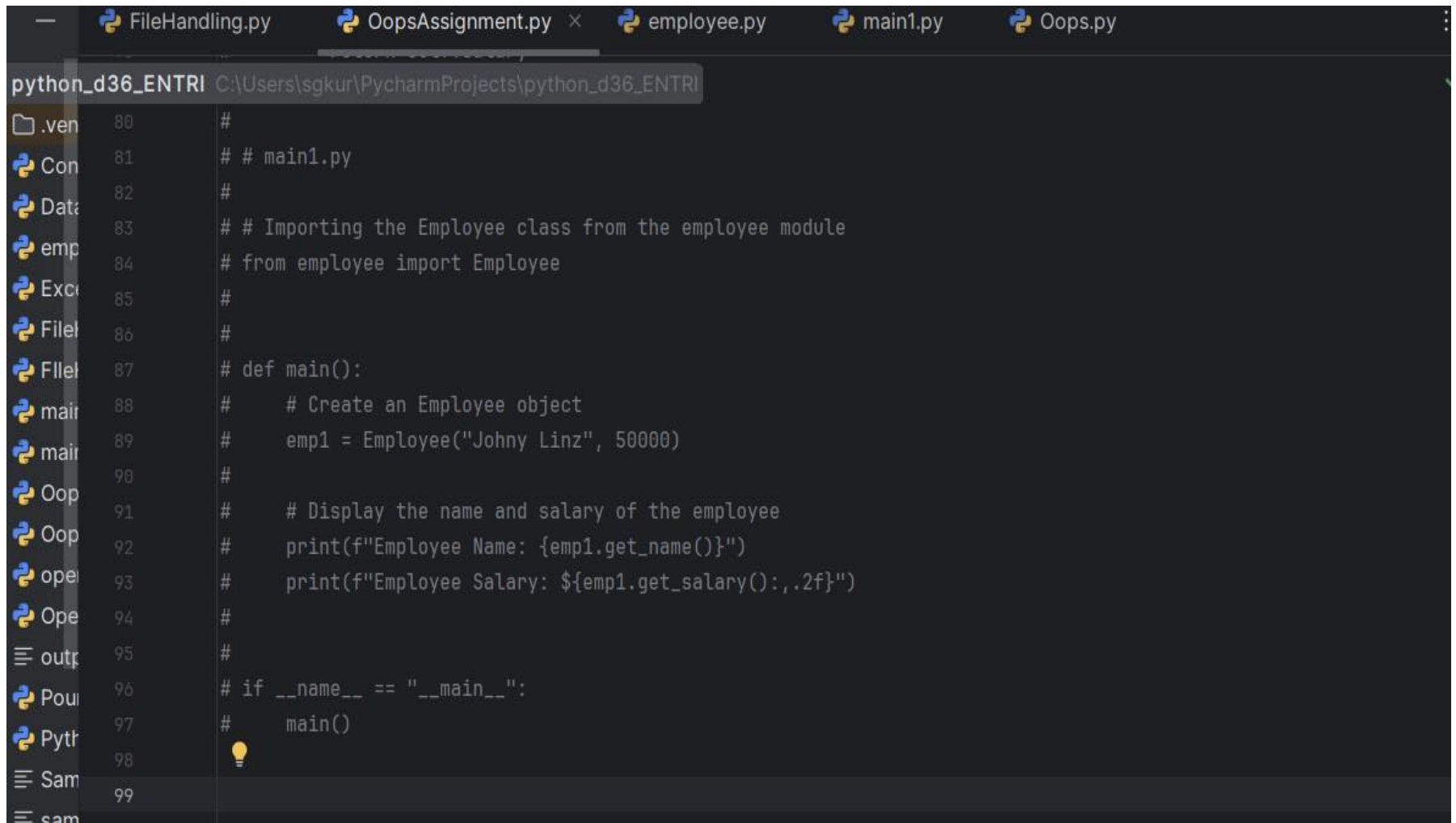


```
60
# 2.Create a Python module named employee that contains a class Employee with attributes name,
# salary and methods get_name() and get_salary(). Write a program to use this module to create an object
# of the Employee class and display its name and salary.

# employee.py module

# class Employee:
#     def __init__(self, name, salary):
#         self.name = name      # Employee's name
#         self.salary = salary  # Employee's salary
#
#     def get_name(self):
#         """Returns the name of the employee."""
#         return self.name
#
#     def get_salary(self):
#         """Returns the salary of the employee."""
#         return self.salary
#
#
```

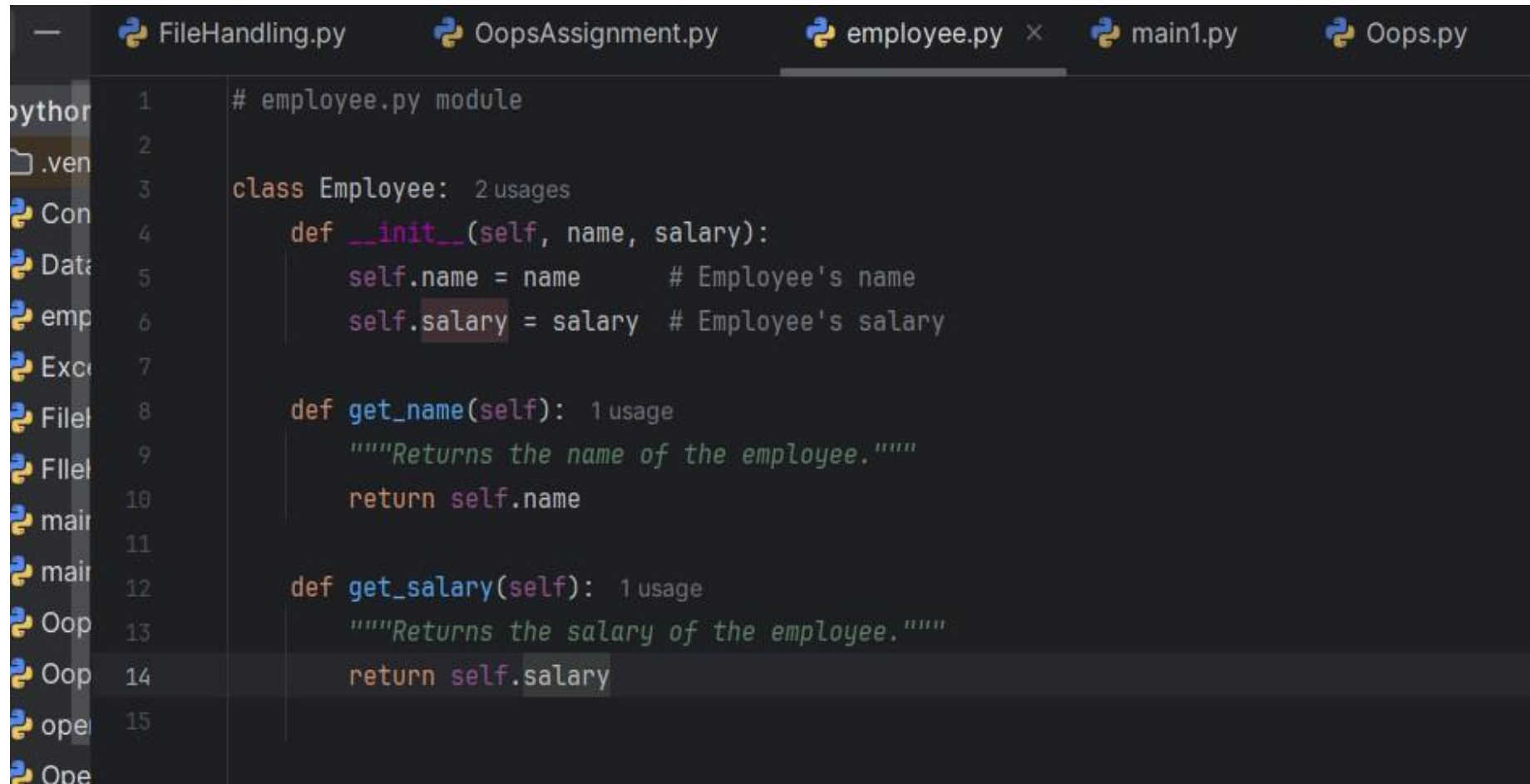

Q.2



```
python_d36_ENTRI C:\Users\sgkur\PycharmProjects\python_d36_ENTRI
FileHandling.py
OopsAssignment.py
employee.py
main1.py
Oops.py

80 #
81 # # main1.py
82 #
83 # # Importing the Employee class from the employee module
84 # from employee import Employee
85 #
86 #
87 # def main():
88 #     # Create an Employee object
89 #     emp1 = Employee("Johny Linz", 50000)
90 #
91 #     # Display the name and salary of the employee
92 #     print(f"Employee Name: {emp1.get_name()}")
93 #     print(f"Employee Salary: ${emp1.get_salary():,.2f}")
94 #
95 #
96 # if __name__ == "__main__":
97 #     main()
98
99
```

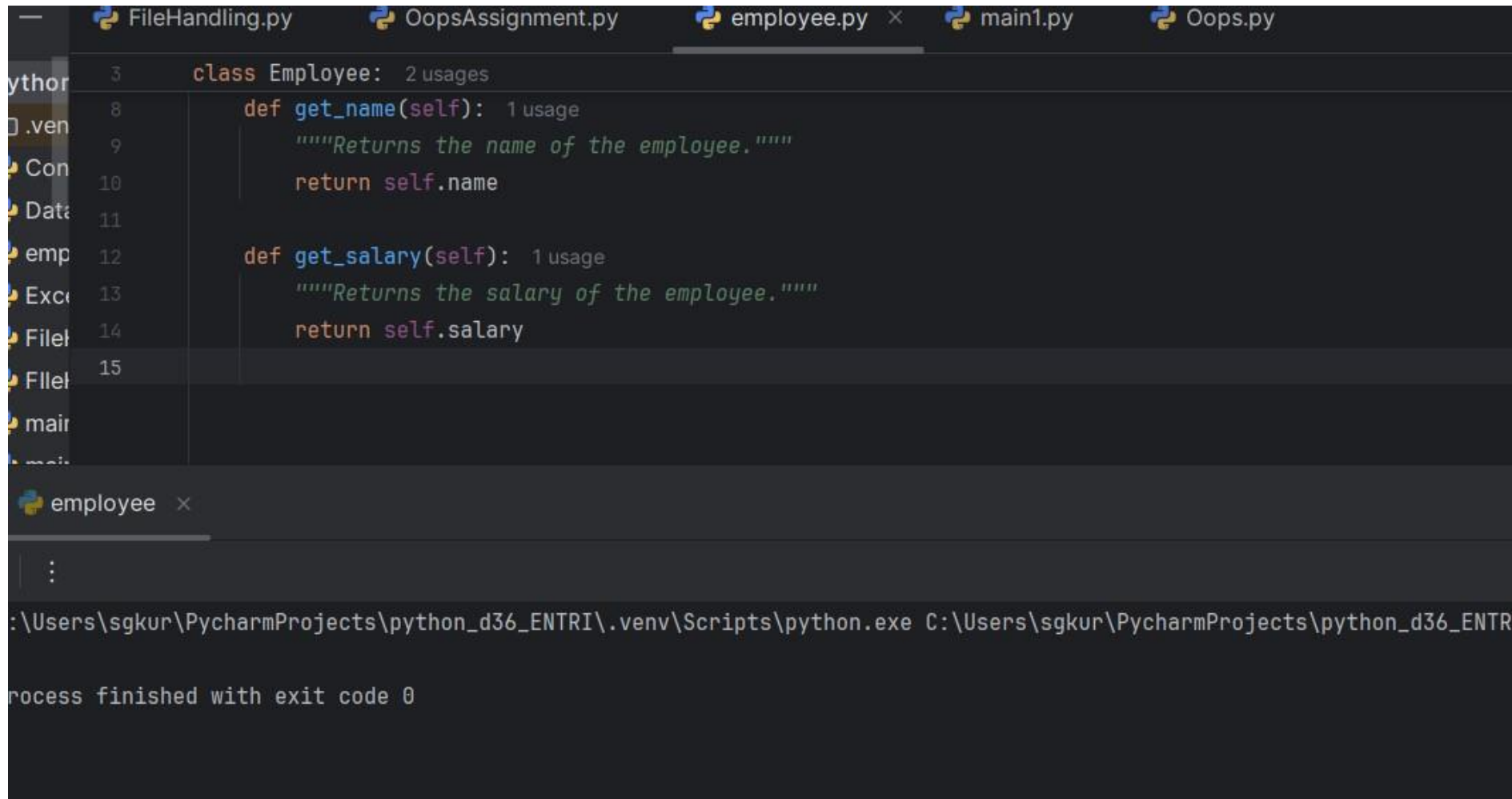

Q.2



The screenshot shows a Python IDE with several tabs open: FileHandling.py, OopsAssignment.py, employee.py (active), main1.py, and Oops.py. The active tab displays the following code:

```
1 # employee.py module
2
3 class Employee: 2 usages
4     def __init__(self, name, salary):
5         self.name = name      # Employee's name
6         self.salary = salary  # Employee's salary
7
8     def get_name(self): 1 usage
9         """Returns the name of the employee."""
10        return self.name
11
12    def get_salary(self): 1 usage
13        """Returns the salary of the employee."""
14        return self.salary
15
```

Q.2



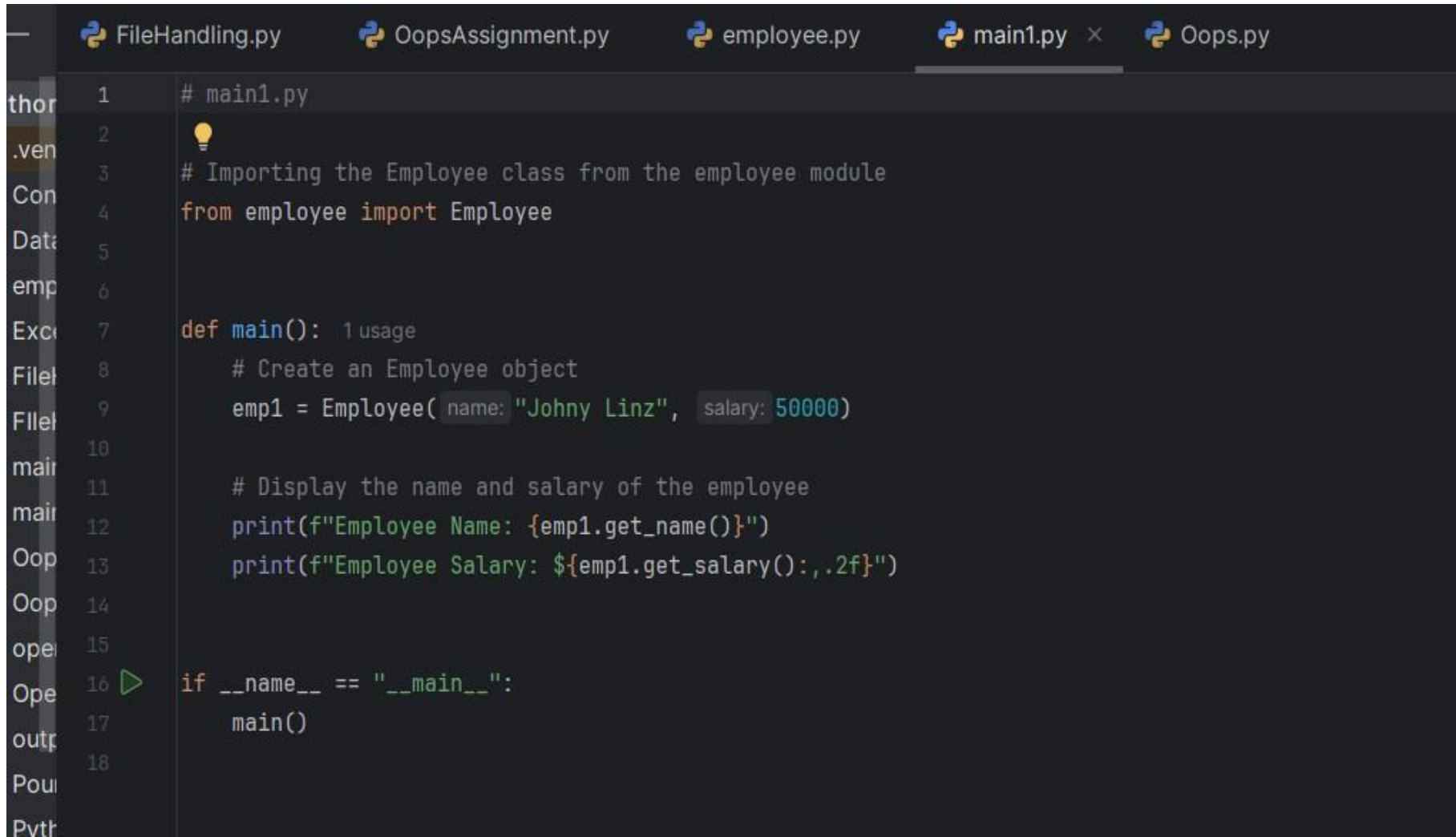
The image shows a PyCharm IDE with several tabs open: FileHandling.py, OopsAssignment.py, employee.py (selected), main1.py, and Oops.py. The editor displays a Python class named `Employee` with two methods: `get_name` and `get_salary`. The `get_name` method returns `self.name`, and the `get_salary` method returns `self.salary`. Both methods include docstrings: `"""Returns the name of the employee."""` and `"""Returns the salary of the employee."""`. The terminal window at the bottom shows the command `python.exe C:\Users\sgkur\PycharmProjects\python_d36_ENTRI` and the message `process finished with exit code 0`.

```
3 class Employee: 2 usages
8     def get_name(self): 1 usage
9         """Returns the name of the employee."""
10        return self.name
11
12    def get_salary(self): 1 usage
13        """Returns the salary of the employee."""
14        return self.salary
15
```


python_d36_ENTRI

process finished with exit code 0

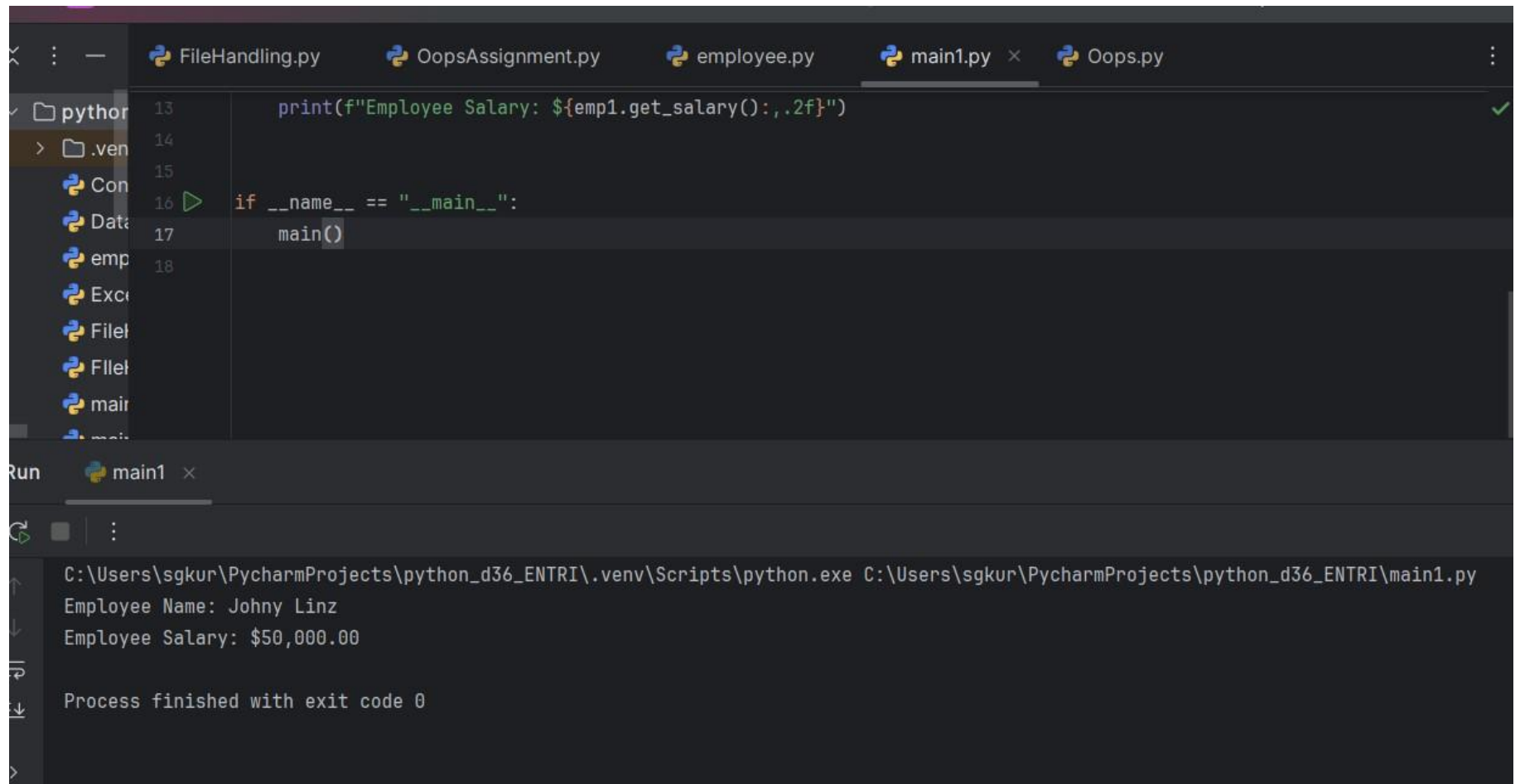
Q.2



```
FileHandling.py  OopsAssignment.py  employee.py  main1.py  Oops.py

1  # main1.py
2  
3  # Importing the Employee class from the employee module
4  from employee import Employee
5
6
7  def main():
8      # Create an Employee object
9      emp1 = Employee(name="Johny Linz", salary=50000)
10
11     # Display the name and salary of the employee
12     print(f"Employee Name: {emp1.get_name()}")
13     print(f"Employee Salary: ${emp1.get_salary():,.2f}")
14
15
16  if __name__ == "__main__":
17     main()
18
```

Q.2



```
FileHandling.py  OopsAssignment.py  employee.py  main1.py  Oops.py

python 13      print(f"Employee Salary: ${emp1.get_salary():,.2f}")
> .venv 14
Con 15
Data 16  if __name__ == "__main__":
emp 17      main()
Exce 18

Run  main1 x

C:\Users\sgkur\PycharmProjects\python_d36_ENTRI\.venv\Scripts\python.exe C:\Users\sgkur\PycharmProjects\python_d36_ENTRI\main1.py
Employee Name: Johny Linz
Employee Salary: $50,000.00

Process finished with exit code 0
```